Application of knowledge-based approaches in software architecture

Li, Zengyang; Liang, Peng; Avgeriou, Paris

Published in:
Information and Software Technology

DOI:
10.1016/j.infsof.2012.11.005

Information and Software Technology 55 (2013) 777–794

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Application of knowledge-based approaches in software architecture: A systematic mapping study

Zengyang Li [a,*], Peng Liang [b], Paris Avgeriou [a]

[a] Department of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands
[b] State Key Lab of Software Engineering, School of Computer, Wuhan University, 430072 Wuhan, China

ARTICLE INFO

Article history:
Received 16 August 2012
Received in revised form 30 October 2012
Accepted 20 November 2012
Available online 5 December 2012

Keywords:
Software architecture
Architecting activity
Knowledge-based approach
Systematic mapping study

ABSTRACT

Context: Knowledge management technologies have been employed across software engineering activities for more than two decades. Knowledge-based approaches can be used to facilitate software architecting activities (e.g., architectural evaluation). However, there is no comprehensive understanding on how various knowledge-based approaches (e.g., knowledge reuse) are employed in software architecture.
Objective: This work aims to collect studies on the application of knowledge-based approaches in software architecture and make a classification and thematic analysis on these studies, in order to identify the gaps in the existing application of knowledge-based approaches to various architecting activities, and promising research directions.
Method: A systematic mapping study is conducted for identifying and analyzing the application of knowledge-based approaches in software architecture, covering the papers from major databases, journals, conferences, and workshops, published between January 2000 and March 2011.
Results: Fifty-five studies were selected and classified according to the architecting activities they contribute to and the knowledge-based approaches employed. Knowledge capture and representation (e.g., using an ontology to describe architectural elements and their relationships) is the most popular approach employed in architecting activities. Knowledge recovery (e.g., documenting past architectural design decisions) is an ignored approach that is seldom used in software architecture. Knowledge-based approaches are mostly used in architectural evaluation, while receive the least attention in architecture impact analysis and architectural implementation.
Conclusions: The study results show an increased interest in the application of knowledge-based approaches in software architecture in recent years. A number of knowledge-based approaches, including knowledge capture and representation, reuse, sharing, recovery, and reasoning, have been employed in a spectrum of architecting activities. Knowledge-based approaches have been applied to a wide range of application domains, among which "Embedded software" has received the most attention.

Contents

* Corresponding author. Tel.: +31 503637127.
E-mail address: zengyangli@gmail.com (Z. Li).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.11.005

1. Introduction

Software architecture (SA) has emerged in the early 1990s as a distinct discipline within software engineering (SE) and entered its golden age a decade later [1,2]. SA is defined in the ISO/IEC IEEE 42010 standard as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" [3]. We adopt this definition for the current study; we further discuss more details about the process of architecting in Section 2.1.1.

Knowledge management (KM) technologies have been employed across the spectrum of SE activities for more than two decades [4,5], for example in requirements elicitation [6], architecture evaluation [7], software testing [8], and software documentation [9]. In recent years, the SA community has paid increasing attention to the application of KM in the architecting process [10], establishing the field of architectural knowledge (AK). This has resulted in acknowledging that the most important types of AK, architectural design decisions (e.g., choosing a particular architectural pattern for a design issue) and design rationale should be treated as first-class entities in SA [11,12].

In the field of AK, a number of KM approaches and tools have been developed for efficiently and effectively improving the practice of software architecting [13,14]. Software architecting is essentially a knowledge-intensive process that is composed of several activities, e.g., architectural analysis, synthesis, and evaluation [15]. Architecting activities are mainly conducted by an architect in collaboration with a set of stakeholders involved towards the construction of the architecture of a software-intensive system. Knowledge-based approaches, such as knowledge sharing and reuse, can be used to facilitate architecting activities. For instance, knowledge about the use of specific architectural patterns in a system development can be shared and reused during architectural synthesis of a similar system. In order to make systematic use of knowledge-based approaches to facilitate architecting activities and develop appropriate methods for using knowledge-based approaches in SA, it is necessary to have a clear understanding on the state of the art of the application of knowledge-based approaches in software architecture. To the best of our knowledge, there is currently no study on what knowledge-based approaches have been employed in what architecting activities, and on the gaps in the existing application of these approaches to various architecting activities. Note that the "application" of knowledge-based approaches in this study does not limit to the practical application of knowledge-based approaches in architecting, but the use of knowledge-based approaches in architecting in both industry and research, for the purpose of a wide coverage of this study. In addition, the comprehensive understanding by this study can also identify the needs for applying specific knowledge-based approaches in architecting and which architecting activities require more exploration on the application of knowledge-based approaches, thereby proposing promising research directions.

To address this issue, we conducted a systematic mapping study (or shortly *mapping study*) on the application of knowledge-based approaches in SA. A systematic mapping study is a form of secondary study aimed at getting a comprehensive overview on a certain research topic, identifying research gaps, and collecting evidence in order to direct future research [16,17]. It allows the studies in a domain to be plotted at a high level of granularity thereby answering broader research questions regarding the current state of the research on a topic [16]. Another form of secondary study is a systematic literature review (SLR), which targets to identify, evaluate, and interpret all available studies to answer particular research questions, which require more in-depth analysis [16]. We selected to perform a mapping study instead of a SLR since the involved domains, i.e., software architecture and knowledge-based approaches, are quite broad areas. Our focus is not on some particular aspects of the involved domains, but the combination of two domains.

The outcomes of this mapping study are the identified studies with different levels of evidence and a classification and a thematic analysis of existing approaches on the application of knowledge-based approaches in SA. This will in turn identify the gaps in the existing application of knowledge-based approaches to various architecting activities. This mapping study was performed following the guidelines for performing SLRs in software engineering proposed in [16]. Although these guidelines are dedicated to performing SLRs, they can also be used for conducting mapping studies if certain deviations are made. The most important difference

between the process of this mapping study and the guidelines in [16], is that study quality assessment was not used as a criterion for study selection. The reason is that study quality assessment is not essential in a systematic mapping study, but all papers related to the topic of the mapping study should be selected. We performed a simplified study quality assessment and considered it as a means of reducing conclusion validity of this mapping study.

The rest of this paper is organized as follows: Section 2 describes the context, research questions, and execution steps of this mapping study. Section 3 presents the synthesis results of the extracted data from the selected studies and answers the research questions. Section 4 discusses the threats to validity of this mapping study. Section 5 comprises a discussion of the results and their implications, with conclusions in Section 6.

2. Mapping study process

The context of this study, including the involved domains (software architecture and knowledge management) and the research questions, is discussed in Section 2.1. The detailed process of this mapping study is presented in Section 2.2.

2.1. Context and research questions

To clarify the scope of this mapping study, several key concepts need to be explained before formulating the research questions. Architecting activities that comprise the architecting process and knowledge-based approaches in knowledge management are the core concepts of this mapping study. We first describe the architecting activities and knowledge-based approaches, and then define the research questions of this mapping study.

2.1.1. Architecting activities

During the architecting process, architects perform various activities for different purposes towards the construction of the architecture of a software-intensive system. Hofmeister et al. defined a general model of architecture design including three activities, i.e., *architectural analysis*, *architectural synthesis*, and *architectural evaluation* [15]. The first one concerns the problem space of architecture design, while the second and third activities concern the solution space. The definitions of these architecting activities are presented below:

- *Architectural Analysis (AA)*: examining, filtering, and/or reformulating architectural concerns and context in order to come up with a set of architecturally significant requirements (ASRs) [15]. This activity aims to define the problems that an architecture needs to address.
- *Architectural Synthesis (AS)*: proposing a collection of architecture solutions to address the ASRs that are identified during AA [15]. This activity essentially links the problem to the solution space.
- *Architectural Evaluation (AE)*: evaluating the candidate architectural solutions that are proposed during AS against the ASRs [15]. This activity helps to detect potential drawbacks on candidate architectural solutions and selects appropriate solutions for ASRs by making decisions and tradeoffs.

Tang et al. extended this general model to five architecting activities in the architecture lifecycle with *architectural implementation* and *architectural maintenance* [14]. The definitions of these two architecting activities are listed below:

- *Architectural Implementation (AI)*: refining architecture design into detailed design, and then implementing the design in code [14]. This activity implements the software system according to the architecture design.
- *Architectural Maintenance and Evolution (AME)*: correcting faults, adapting to a changed or changing operational environment, and implementing new requirements in an architecture [18,19]. This activity ensures the consistency and integrity of the architecture during the next product releases.

Please note that we have combined architectural maintenance and evolution in the same activity. We adopt the following definitions: architectural maintenance is about correcting faults and adapting to a changed or changing operational environment [18], while architectural evolution focuses on implementing new requirements [19,20].

The five architectural activities mentioned above cover the entire architecture lifecycle. We call them specific architecting activities to distinguish them from general activities described later. General architecting activities can be performed across the architecting process to support architects in achieving the goals of specific architecting activities. We identified the following general architecting activities:

- *Architecture Recovery (AR)*: uncovering architecture design and related design decisions based on existing implementation and documentation of the system [21]. This activity transforms existing architectures from implicit to explicit.
- *Architecture Description (ADp)*: documenting an architecture using a collection of elements, such as architecture views and models. This activity codifies architecture in a consistent way. The main goals of ADp are facilitating the expression and evolution of software systems, providing a blueprint for system development, and supporting the communication among stakeholders [19].
- *Architecture Understanding (AU)*: comprehending the elements of an architecture design and their relationships, as well as the corresponding design decisions (why the architecture is designed the way it is). This activity helps architects and concerned stakeholders to acquire thorough knowledge about an architecture. The difference between AR and AU, is that the information made explicit during AR is used as input to AU and gets transformed into knowledge.
- *Architecture Impact Analysis (AIA)*: identifying the architectural elements affected by a change scenario, including directly-affected and indirectly-influenced parts of an architecture [22]. The outcome of this activity helps architects understand the dependencies between the changed parts and the affected parts of an architecture.
- *Architecture Reuse (ARu)*: using existing architectural assets, such as architectural design elements, decisions, patterns, and other assets, in the solutions addressing various architecting problems [23]. By performing this activity, architectures of better quality can be achieved at lower cost.

Note that, as described in Section 2.2.1.2, we conducted a trial study search before the formal study search of this mapping study. These general architecting activities were identified according to the results of the trial study search. We started with this initial list, with the option of adding other general architecting activities identified during the search process. Eventually, we did not identify other general architecting activities during the formal study search.

As mentioned before, the general architecting activities can be used to facilitate specific architecting activities during the architecting process. Based on our own understanding and experience on the architecting activities, we outline what the primary support that each general architecting activity can provide to a specific

Table 1
Primary support from general to specific architecting activities.

General architecting activity	Specific architecting activity				
	AA	AS	AE	AI	AME
AR					✓
ADp		✓			
AU	✓		✓	✓	✓
AIA	✓	✓			✓
ARu		✓			

architecting activity in Table 1. AR can support evolving a legacy system or a system without a well-documented architecture during AME; ADp can be used to record the architecture design and related decisions with the rationale behind them during AS; AU can help architects to get a fair understanding for analyzing, evaluating, implementing, maintaining, and further evolving a system (i.e., support for AA, AE, AI and AME); AIA can assist architects to identify affected architectural elements when a new design decision is added or an existing design decision is changed during AS and AME, and when analyzing if a new requirement is an ASR (i.e., support for AS, AME, and AA); and ARu is performed to use the existing architectural assets to address similar problems during AS. Note that other types of secondary support may exist from general to specific activities but are not discussed here. Our goal here is not to exhaust all possible types of support from general to specific architecting activities but to present and explain the potential support from general to specific architecting activities.

2.1.2. Knowledge-based approaches

A knowledge-based approach in this study refers to any approach from KM that can directly contribute to the architecting process. We identified five such approaches, i.e., knowledge capture and representation, reuse, sharing, recovery, and reasoning, from two previous survey papers on AK [13,14]. These approaches are described below.

- *Knowledge Capture and Representation* (KCR) extracts knowledge from diverse sources as well as its acquisition directly from the stakeholders, and expresses knowledge in certain forms so that the knowledge can be used for automatic or human reasoning. On one hand, knowledge representation accompanies knowledge capture since the knowledge should be represented in certain form when captured; on the other hand, knowledge capture may happen during knowledge representation. For instance, when one uses a conceptual model to transform (e.g., annotate) implicit knowledge to explicit knowledge, knowledge capture takes place at the same time. We thus consider knowledge capture and knowledge representation as a combined knowledge approach.
- *Knowledge Reuse* (KR) applies existing knowledge (e.g., architectural patterns) in a particular context for various purposes [24].
- *Knowledge Sharing* (KS) exchanges knowledge (e.g., skills or expertise) among individuals in a community or an organization.
- *Knowledge Recovery* (KRv) recovers explicit knowledge from tacit knowledge, e.g., decision rationale that is not documented. In KM theory, knowledge is classified into two types: tacit knowledge which resides in people's head, and explicit knowledge which is codified in a certain form [25].
- *Knowledge Reasoning* (KRs) draws conclusions and derives new knowledge from existing knowledge through inference. An example is reasoning based on the rationale knowledge of the existing architectural design decisions to make new decisions addressing new or modified design issues.

The aforementioned knowledge-based approaches can be mapped on the knowledge activities in a KM framework proposed by Maryam Alavi et al. to analyze KM systems [26]. This framework includes four basic knowledge activities: knowledge creation, knowledge storage/retrieval, knowledge transfer, and knowledge application. In this paper, we do not consider knowledge retrieval, as it does not directly contribute to the architecting process: it concerns the methods of knowledge access and is thus a supporting activity for the other knowledge activities. The rest of the basic knowledge activities are regarded to make direct contribution to the architecting process. First, new knowledge is created when the architect devises solutions for architecture problems. Second, knowledge storage preserves various types of knowledge on architecture in a certain form. This facilitates the reasoning on existing knowledge to solve new architecture issues and supports the reuse of the existing knowledge to address similar issues. Third, knowledge transfer is a prerequisite for knowledge application. Existing knowledge (e.g., best practices) needs to be transferred to the architect in order to be applied to solve a problem. Finally, knowledge application is to (re)use the existing knowledge to tackle architecture issues. Each knowledge-based approach can be mapped on one of the basic knowledge activities in the KM framework in [26] according to its functionality (see Table 2).

2.1.3. Research questions

The overall goal of this mapping study is to get an overview of existing research on the application of knowledge-based approaches introduced in Section 2.1.2 in architecting activities. To obtain a detailed and comprehensive view on this topic, this high-level goal is decomposed into four research questions (RQs):

RQ1: Which knowledge-based approaches are applied in the architecting activities?

> *Rationale*: Architecting is a knowledge-intensive process, and different knowledge-based approaches have been used in this process. We want to know which knowledge-based approaches are used and how often each knowledge-based approach is used in the architecting activities. This information can help us to identify the gaps of the application of various knowledge-based approaches and the needs for applying specific knowledge-based approaches.

RQ2: In which architecting activities have knowledge-based approaches been applied?

> *Rationale*: Various architecting activities may have specific needs on different knowledge-based approaches. We want to know in which architecting activities the knowledge-based approaches have been applied and how often the knowledge-based approaches have been applied in each architecting activity. This information can help us to identify the gaps of the application of knowledge-based approaches over various architecting activities and to identify which architecting activities require more exploration on the

Table 2
Mapping from knowledge-based approaches to knowledge activities.

Knowledge-based approach	Knowledge activity in the KM framework in [26]
Knowledge Capture and Representation (KCR)	Knowledge storage
Knowledge Reuse (KR)	Knowledge application
Knowledge Sharing (KS)	Knowledge transfer
Knowledge Recovery (KRv)	Knowledge creation
Knowledge Reasoning (KRs)	Knowledge creation

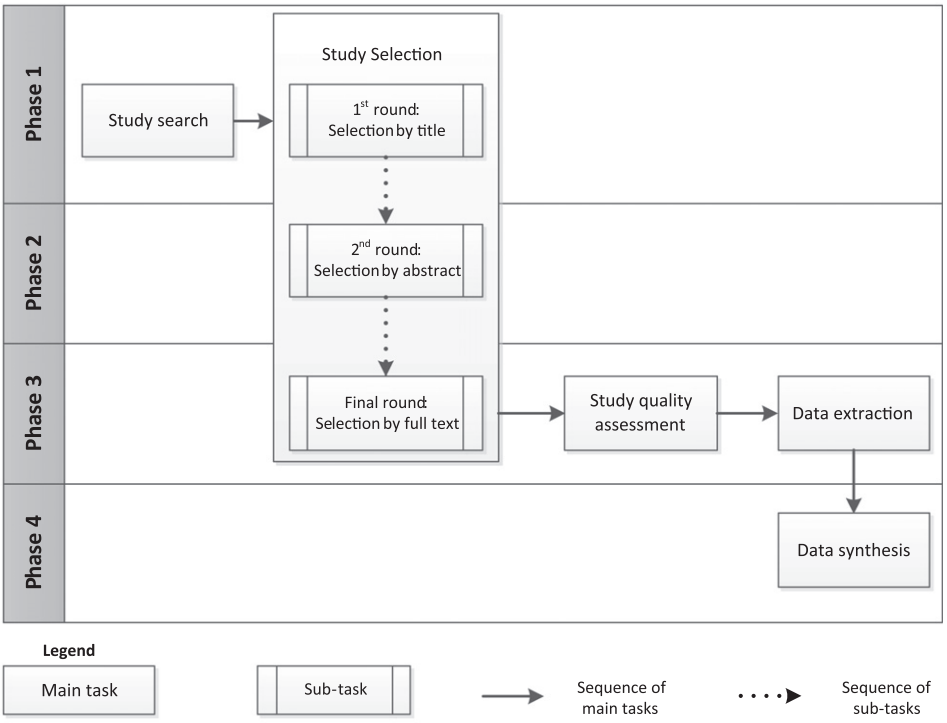

Fig. 1. Design of the systematic mapping study.

application of knowledge-based approaches. The difference between RQ1 and RQ2 is that the former is centered around knowledge-based approaches, while the latter is centered around architecting activities.

RQ3: In which sources and in which years have the studies of the application of knowledge-based approaches in the architecting activities been published?

Rationale: The topic of this mapping study is broad and there should be a number of venues to publish the related studies. In addition, the research on the study topic has been carried out for decades. We want to get an understanding on whether there are specific publication venues for these studies and when the effort regarding the study topic was made. This information can help us to identify the leading publication venues where the authors can better disseminate their research results and the trend of the number of published studies in this topic.

RQ4: In which application domains have knowledge-based approaches been applied to architecting activities?

Rationale: The use of knowledge-based approaches in architecting activities may take place in various application domains, such as embedded systems and financial software. By answering this question, we want to know what application domains the knowledge-based approaches have been applied to and how often the knowledge-based approaches have been applied to each domain. This information can help us to identify which application domains have gained more interest in applying knowledge-based approaches.

2.2. Mapping study execution

This section describes the process of this mapping study, which is largely based on the guidelines for performing SLRs proposed in [16], but with a deviation. As mentioned in the Introduction Section, the quality assessment of the selected studies was not used as a criterion for study selection.

This mapping study covered five main tasks across four phases as illustrated in Fig. 1. The task of study selection includes three sub-tasks, i.e., selection by title (the first round selection), selection by abstract (the second round selection), and selection by full text (the final round selection). The next round study selection refined the results of the previous round study selection. In phase 1, the study search was performed and, meanwhile, the study selection by title was conducted. In phase 2, we did the study selection by abstract based on the results of the study selection by title. In phase 3, we read the full text of all the papers filtered in the study selection by abstract. To make the review more efficient, we simultaneously conducted three (sub-) tasks: the study selection by full text, study quality assessment, and study data extraction. This means, when reading a paper, we made the final decision on whether a paper should be selected or not. If this paper was finally selected, we assessed its quality and extracted the data from it. In this way, we only needed to read the paper once for performing the three (sub-) tasks. We synthesized the extracted data in phase 4 to answer the research questions. The following subsections elaborate on the main tasks of this mapping study.

2.2.1. Study search

We employed two search methods: the automatic search and the manual search. With the automatic search method, we searched studies in an electronic database (e.g., IEEEXplore) with search terms by the search engine provided by this database. The search engine checks the search terms against the metadata, including the title, abstract, and keywords, of each paper in the database. With the manual search method, we browsed all the papers within the time period specified in Section 2.2.1.1 in target venues (i.e., the journals, conferences, and workshops listed in Tables 4–6, respectively) without using search terms but by using the names of the target venues. The use of an automatic search ensures a high coverage (i.e., retrieving studies in as many venues as

possible) of potentially-relevant studies by searching electronic databases using search terms. On the other hand, it may provide many irrelevant studies. Manual search aims at increasing the completeness (i.e., deriving as many studies as possible in target venues) of potentially-relevant studies in target journals, conferences, and workshops, which are of high relevance to the study topic. Although most of the target venues are indexed by the included databases, the manual search is complementary instead of repetitive to the automatic search: the relevant studies published in the target venues may be filtered out in the automatic search, but are likely to be found in the manual search. Moreover, some target venues (e.g., SEKE and IJSEKE) are not indexed in the included databases, and the manual search therefore may retrieve some relevant studies that the automatic search cannot find. The results of manual search are independent of the results of the automatic search, thus these two search methods can be performed in an arbitrary sequence in this mapping study. In the rest of this section, we first scope the time period and publication sources of the study search, and then present the search strategy used in this mapping study.

2.2.1.1. Search scope. This subsection describes the search scope of this mapping study, including time period and search sources. We include a number of electronic databases (listed in Table 3), journals (listed in Table 4), conferences (listed in Table 5), and workshops (listed in Table 6) that are relevant to the study topic.

- **Time period**

We scope the time period of the related studies published from January 2000 to March 2011. March 2011 is the end time of the period since it was the time that we started this mapping study. According to the description in [2] by Shaw and Clements, software architecture started to become popular around the year 2000. Therefore, we chose January 2000 as the starting time of the period.

- **Electronic databases**

As shown in Table 3, six electronic databases were included in this mapping study. According to the results in [27], these six databases are the most popular ones in computer science and engineering that ensure a high coverage of potentially-relevant studies. Even though EI Compendex is considered as a very important source, it was not included as it is not accessible in our universities. In addition, many venues indexed by EI Compendex are also indexed by other databases. We did not include Google Scholar since the search results of Google Scholar tend to be repetitive with the search results from the included electronic databases, and its unique contribution to study search is unclear [27].

- **Journals, conferences, and workshops**

A set of journals (Table 4), conferences (Table 5), and workshops (Table 6) that are of high relevance with the study topic were included. Two criteria for selecting the publication venues are that (1) they should be highly relevant to SA or both SA and KM and (2) they are leading journals, conferences, and workshops in the study area. Conferences such as C3, C4, and C5 and workshop W1 were reasonably chosen as the sources in this mapping study since they are top conferences and unique workshop on SA and knowledge-based SA. As SA is a significant sub-area of software engineering (SE), papers on SA are usually published in journals and conferences in SE. Therefore, the publication venues in SE or both SE and KM should be included. Such publication venues include journals J1, J2, J3, J4, J5, J6, J7, J8, J10, J11, and conferences C1 and C2. Workshop W2 was included because it is a workshop about requirements knowledge, which has a close relationship to architectural analysis, especially in terms of architecturally significant requirements. Conferences C6 and C7 were selected because studies on architectural maintenance and evolution can be published in these two conferences. In addition, journal J9 focuses on software quality where research on knowledge-based approaches to improve architecture quality can be published.

Table 3
Electronic databases included in the automatic search of the mapping study.

#	Electronic Database
DB1	IEEE Xplore
DB2	ACM Digital library
DB3	Science Direct
DB4	ISI Web of Science
DB5	SpringerLink
DB6	INSPEC

Table 4
Journals included in the manual search of the mapping study.

#	Journal
J1	IEEE Transactions on Software Engineering (TSE)
J2	Empirical Software Engineering (ESE)
J3	IEEE Software (IEEE SW)
J4	International Journal of Software Engineering and Knowledge Engineering (IJSEKE)
J5	Journal of Systems and Software (JSS)
J6	Information and Software Technology (IST)
J7	Software Process Improvement and Practice (SPIT)
J8	Software and System Modeling (SoSyM)
J9	Software Quality Journal (SQJ)
J10	Automated Software Engineering (ASE)
J11	Software: Practice and Experience (SPE)

Table 5
Conferences included in the manual search of the mapping study.

#	Conference
C1	Software Engineering and Knowledge Engineering (SEKE)
C2	International Conference on Software Engineering (ICSE)
C3	The Working IEEE/IFIP Conference on Software Architecture (WICSA)
C4	European Conference on Software Architecture (ECSA)
C5	International Conference on the Quality of Software Architectures (QoSA)
C6	International Conference on Software Maintenance (ICSM)
C7	European Conference on Software Maintenance and Reengineering (CSMR)

Table 6
Workshops included in the manual search of the mapping study.

#	Workshop
W1	SHAring and Reusing architectural Knowledge (SHARK)
W2	Managing Requirements Knowledge (MaRK)

2.2.1.2. Search strategy. Search strategy is important for a mapping study, since it affects the quality and completeness of retrieved studies and the effort we need to spend. The search strategy and steps in this mapping study are described below:

Step 1: The preliminary search terms were identified according to the study topic and the research questions. The search terms used to retrieve relevant studies in the automatic

search were formed in Steps 1 and 2. Similar in [28], we adopted the PICO (Population, Intervention, Comparison, Outcomes) criteria to formulate the search terms. Population includes the terms about architecture; intervention includes knowledge-related terms; and outcomes include the terms relevant to architecting activities. The comparison part is not applicable in this mapping study because this mapping study does not involve the comparison of the knowledge-based approaches and other types of approaches [16].

Step 2: Trial searches using various combinations of search terms were performed. The search terms were improved according to the results of the trial searches. The trial results inspired us to come up with some missing terms and synonyms of existing terms. Both types of terms were added when performing formal searches. We got the terms included in the three parts listed below:

Population: architecture, architectural, architecting.

Intervention: knowledge, semantic, rationale, reasoning, decision, information.

Outcomes: architecture design, architectural design, architecture analysis, architectural analysis, architecture synthesis, architectural synthesis, architecture evaluation, architectural evaluation, architecture maintenance, architectural maintenance, architecture implementation, architectural implementation, architecture recovery, architectural recovery, architecture documentation, architectural documentation, architecting process, architectural preservation, architecture preservation, architectural variability, architecture variability, architecture description, architectural description, architecture understanding, architectural understanding, impact analysis, architecture evolution, architectural evolution.

We used Boolean operator OR to join alternate words and synonyms in each part (i.e., population, intervention, outcomes), and used Boolean operator AND to join the terms from the three parts respectively. The full search string is like this:

(architecture **OR** architectural **OR** architecting) **AND** (knowledge **OR** semantic **OR** rationale **OR** reasoning **OR** decision **OR** information) **AND** ("architecture design" **OR** "architectural design" **OR** ...[each term in the Outcomes]).

To explain the use of the terms and Boolean operators in actual searches, we take the advanced search in IEEEXplore database as an example. We fill in the first field (i.e., the first textbox) in the IEEEXplore search interface with the string "*architecture* ***OR*** *architectural* ***OR*** *architecting*" and the second field with the string "*knowledge* ***OR*** *semantic* ***OR*** *rationale* ***OR*** *reasoning* ***OR*** *decision* ***OR*** *information*". We divide the terms in Outcomes into small groups with two terms each and fill in the third field with one group at a time. The terms in a group are joined by the Boolean operator OR. The three fields are joined by AND. The reason we partition the Outcome terms into small groups is that there are many terms in Outcomes, and the returned results will be too many to judge in a reasonable length of time if we fill in the third field with the long string covering all the terms in Outcomes. In other words, a search in IEEEXplore database is composed of a number of small searches. Note that, generally, a search field in the search interface of a specific database has a limitation on the string length or the number of search terms. Thus, a search is often divided into a number of small searches.

Step 3: Formal searches, including automatic and manual searches, were conducted. Meanwhile, the first round study selection was performed accompanying the study searches. The study selection criteria are specified in Section 2.2.2.1. The results of the first round selection of the two types of searches were recorded, and then were merged after removal of duplicated selection results as the input for the second round study selection.

2.2.2. Study selection

To make the study selection results objective, we defined selection criteria that were employed in the study selection process. In addition, in the second and final round study selections, two reviewers conducted the selection in parallel and harmonized their selection results to mitigate the personal bias in selection results caused by individual reviewers.

2.2.2.1. Selection criteria. The inclusion and exclusion criteria are used in the three rounds of study selections to decide whether a study should be included or not.

Inclusion criteria

I1. The paper is about architectures of software-intensive systems rather than hardware or building architecture. (Some papers on hardware or building architecture may be retrieved in the results of automatic searches.)

I2. The paper employs at least one knowledge-based approach to address the problems in architecting activities.

Exclusion criteria

E1. The paper, in which none of knowledge-based approach is applied or software architecting is not the focus, is excluded.

E2. The paper that is published in the form of abstract, tutorial, or talk is excluded.

Before the study selection, we conducted a pilot selection to ensure that all the reviewers had a consensus on the understanding of these selection criteria. The inclusion criterion I1 is applied in the first and second rounds of study selection but not in the final round since the abstract of a paper has provided enough information to judge whether the paper is really about software architecture or not. The exclusion criterion E2 is not applicable in the first round of study selection as we cannot know the publication type of a paper, i.e., published as a full paper, abstract, tutorial, or talk, by reading its title. The inclusion criterion I2 and exclusion criterion E1 are applicable and used in all the three rounds of study selection.

2.2.2.2. First round study selection. As illustrated in Fig. 1, study selection is divided into three sub-tasks and each sub-task was performed in different phases. As mentioned in the second paragraph of Section 2.2, we performed the first round study selection along with study search. We checked the title of each paper against the inclusion criteria I1 and I2, and the exclusion criterion E1. If there was any doubt whether a paper was relevant or not, it was included for the next round selection.

2.2.2.3. Second round study selection. We read the abstracts of the papers that were left after the first round selection and checked them against the inclusion criteria I1 and I2, and the exclusion criteria E1 and E2. Selection results were verified by two reviewers and any disagreements among the reviewers were discussed and resolved. If a resolution of the disagreement is impossible to achieve at this stage, the paper was included. If it is difficult to judge whether a paper should be included or not, the paper was included for the next round selection.

Table 7
Questions on study quality assessment.

#	Questions
Q1	How much evidence of the proposed method is available?
Q2	Is there a clear statement of the aims of the research?
Q3	Is there a clear statement of what the knowledge-based approach is?
Q4	Is there an adequate description of what architecting activity the knowledge-based approach is applied to?
Q5	Are the limitations of the study discussed explicitly?

2.2.2.4. Final round study selection. We read the full text of the papers left by the second round selection and employed the inclusion criterion I2 and the exclusion criteria E1 and E2 to decide whether the papers will be finally included. In this round of study selection, we also checked whether multiple selected papers published the same study. If two papers publish the same study in different venues (e.g., in a conference and a journal, respectively), the mature one gets selected, and the less mature one (i.e., the one published in a conference) is excluded. Again, we discussed any disagreements on selection results and achieved consensus on the final selection results.

2.2.3. Study quality assessment

Quality assessment of all selected studies is important for the quality of data extraction and synthesis of the mapping study results. All the finally-selected studies underwent such an assessment against a set of questions regarding the evidence level of the study and the quality of the data items to be extracted. Table 7 presents the quality assessment questions used in this mapping study. Question Q1 evaluates the evidence level of the proposed method of the selected study. We adopted the evidence hierarchy proposed in [29]. More specifically, the evidence hierarchy is defined as follows (from weakest to strongest):

1. No evidence.
2. Evidence obtained from demonstration or working out toy examples.
3. Evidence obtained from expert opinions or observations.
4. Evidence obtained from academic studies, e.g., controlled lab experiments.
5. Evidence obtained from industrial studies, e.g., causal case studies.
6. Evidence obtained from industrial practice.

Q2 and Q5 are adopted from [30,31] while Q3 and Q4 are formulated according to our study topic and RQs. Similar to [31], we adopted and adjusted the quality assessment instrument used by Dybå and Dingsøyr [30]. This instrument, i.e., the grading rules for the quality assessment questions of a study, uses a three-point scale to answer each of the last four questions, i.e., "yes", "to some extend", and "no". Each quality assessment question was further quantified by assigning a numerical value to each answer ("yes" = 1, "to some extend" = 0.5, and "no" = 0). For question Q1, a six-point scale is used to grade the six evidence levels (from weakest to strongest evidence, the score of the evidence level of each selected study can be 0.0, 0.2, 0.4, 0.6, 0.8, or 1.0). Then, a quality assessment score can be given to a study by summing up the scores to all the questions for a study.

2.2.4. Data extraction

To answer the RQs defined in Section 2.1.3, we extracted specific data from the selected studies. Table 8 describes the data items to be extracted in this mapping study. The first five data items (D1–D5) and the data item D7 directly contribute to the answers of the RQs. The data item D6 supports the explanation of the results of this mapping study in Sections 3.2 and 3.3. To make sure the data extraction results less biased, two reviewers performed the data extraction independently, and then each reviewer checked the data extraction results by the other reviewer, and finally two reviewers achieved a consensus on the data extraction results.

2.2.5. Data synthesis

Data synthesis targets to synthesize the extracted data to answer the RQs. This task will be detailed in Section 3.

Table 8
Extracted data items.

#	Item Name	Description	Relevant RQ
D1	Source	In which venue was the study published?	RQ3
D2	Publication type	In which publication type was the study published?	RQ3
D3	Publication year	In which year was the study published?	RQ3
D4	Architecting activity/ activities	Which architecting activity/activities (i.e., the activities introduced in Section 2.1.1) does the proposed approach in the study try to improve by knowledge-based approach(es)?	RQ2
D5	Knowledge-based approach(es)	Which knowledge-based approach(es) (i.e., the approaches introduced in Section 2.1.2) are employed in the study to address the problem(s) in architecting activity/activities?	RQ1
D6	Problem(s) addressed	What problem(s) (e.g., poor effectiveness of AIA) in the architecting activity/ activities does the study try to address by the application of knowledge-based approach(es)?	RQ1 RQ2
D7	Application domain	What application domain (e.g., embedded software) is the knowledge-based approach(es) applied to in this study?	RQ4

3. Study results

We performed the mapping study according to the steps described in Section 2.2. We first present an overview of the study results, and then analyze the results of this mapping study to answer the RQs defined in Section 2.1.3.

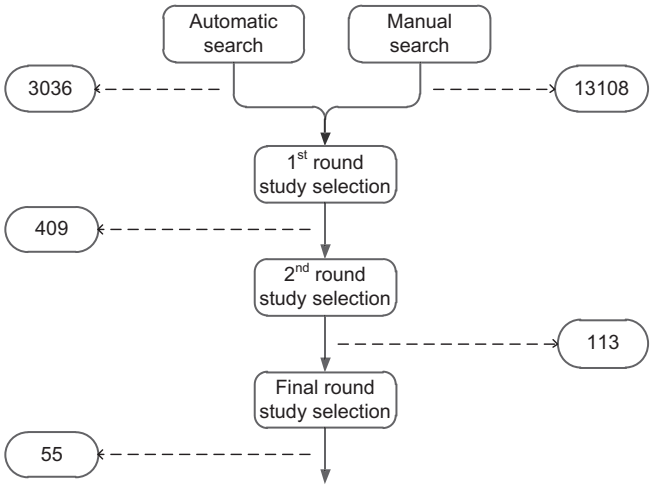

Fig. 2. Study search and selection results in three rounds of selection.

3.1. Overview of results

The study search and selection results, and the study results distribution are presented in this section.

3.1.1. Search and selection results

Fig. 2 presents the study search and selection results in the three rounds of selection. In the search phase, we retrieved 16,144 papers, including 3036 papers from the automatic searches in electronic databases and 13,108 papers from the manual searches in target journals, conferences, and workshops (i.e., all the papers published in the target sources between 2000 and 2011). 409 studies were left after the first round study selection. 296 studies were excluded in the second round study selection and 113 studies were left for the final round study selection. Finally, we got 55 studies selected.

3.1.2. Study results distribution

By analyzing the data extracted from the 55 selected studies, we get an overview of the study results distribution. As shown in Fig. 3, a map is used to present the results distribution on the study topic "Application of Knowledge-based Approaches in Software Architecture (AKASA)" over the range of architecting activities, knowledge-based approaches, and time period. In the left part of Fig. 3, a bubble represents one study or several studies on an architecting activity published in a certain year. In the right part of Fig. 3, a bubble denotes one study or several studies applying a certain knowledge-based approach in an architecting activity. The numbers in a bubble are the identification numbers of the corresponding studies in the 55 selected studies as listed in Appendix A. For instance, the left-top bubble in the right part of Fig. 3 denotes that the 6th, 40th, and 48th studies are about the application of knowledge reuse in the architecture reuse activity. In the rest of the paper, study *Sn* denotes the *n*th study in the 55 selected studies. The detailed analysis of this map (Fig. 3) is presented in the rest of this section.

3.2. Knowledge-based approaches in architecting activities

This subsection gives the answer about which knowledge-based approaches are applied in various architecting activities.

Looking at the map (Fig. 3) from the perspective of knowledge-based approach dimension, we get the distribution of selected studies over knowledge-based approaches in Fig. 4. This figure

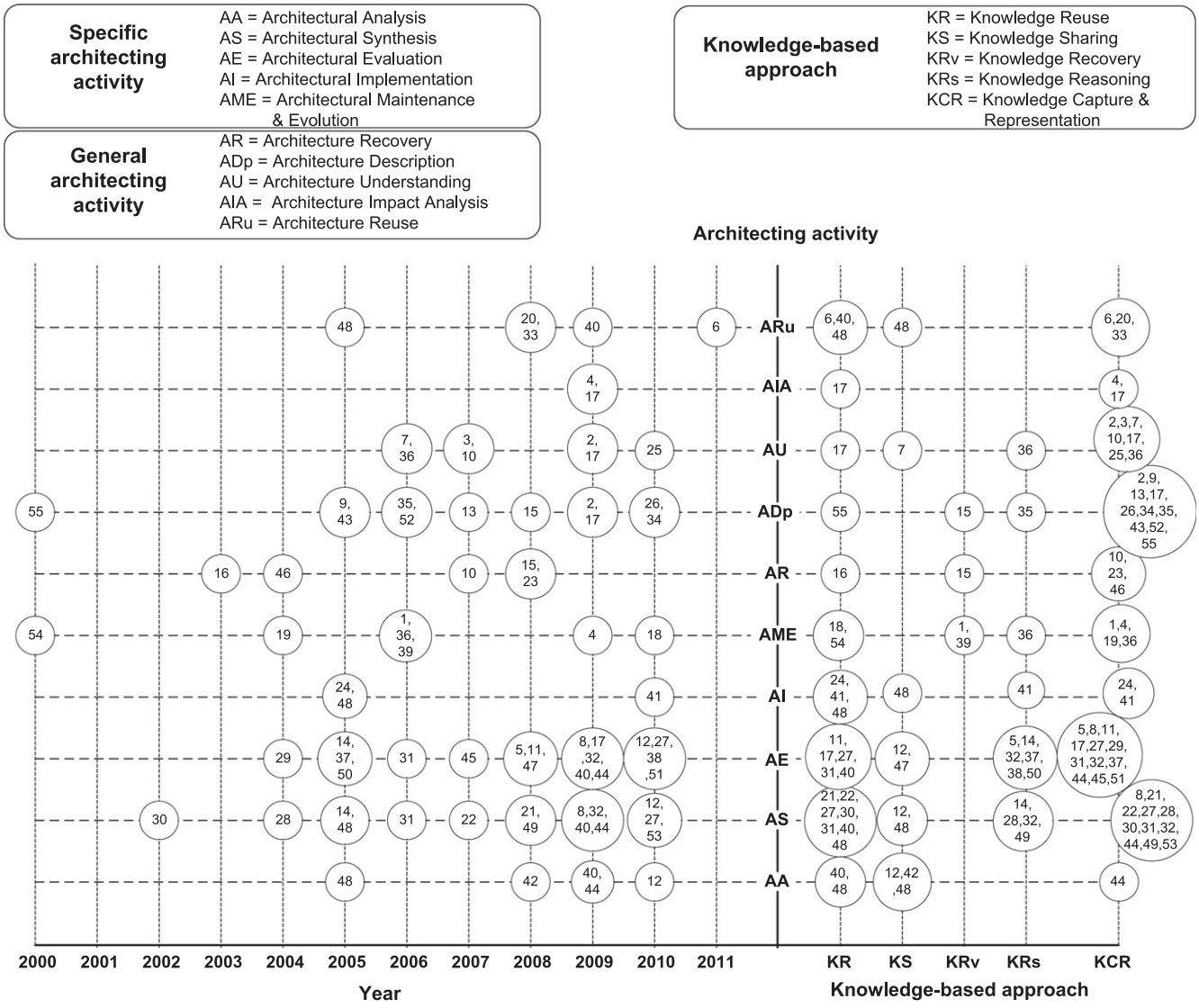

Fig. 3. Studies distribution of AKASA over the range of architecting activities, knowledge-based approaches, and time period.

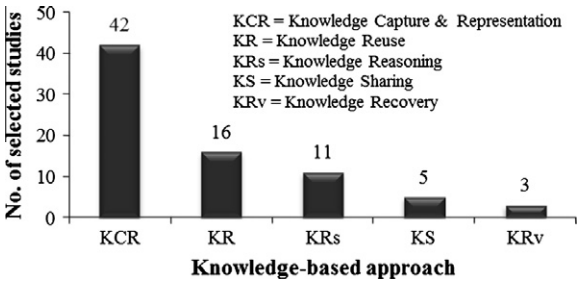

Fig. 4. Distribution of selected studies over knowledge-based approaches.

shows that knowledge-based approaches KCR, KR, KS, KRs, and KRv are all used in architecting activities, but there is a significant difference in terms of the popularity of these knowledge-based approaches.

Knowledge Capture and Representation (KCR) is most frequently used in architecting activities among the five knowledge-based approaches. KCR is used in 42 studies out of total 55, which means 76.4% of the total studies employ this knowledge-based approach in various architecting activities. Knowledge Recovery (KRv) and Knowledge Sharing (KS) are seldom used. More specifically, KRv is employed in only three studies out of total 55 (i.e., 5.5%), and KS is used in five studies out of total 55 (i.e., 9.1%). Knowledge Reuse (KR) and Knowledge Reasoning (KRs) rank second and third respectively among the knowledge-based approaches in terms of the popularity. Comparing with KCR, these two approaches are merely employed in a small number of studies, but these two approaches are used more than twice as much as KS and KRv.

Knowledge sharing is seldom employed in architecting activities. In the study selection process, we did retrieve a number of papers that mentioned knowledge sharing, but most of them are about AK sharing tools (such as [32–34]), how to better share knowledge for certain purposes (e.g., [35]), or using various knowledge sharing models (e.g., [36,37]), instead of how to improve architecting activities with knowledge sharing.

There are only three selected studies using knowledge recovery to support architecting activities. To be specific, *S1* uses semantic annotation, which is a method of knowledge recovery, to facilitate refactoring architecture artifacts for maintaining the system quality during AME; *S15* proposes Architectural Design Decision Recovery Approach (ADDRA) to recover architectural design decisions for the purpose of improving architecture documentation (note that ADDRA is also a method for architecture recovery); *S39* presents an approach to recover implicit assumptions in architecture design to support AME.

Fig. 5 shows the distribution of the selected studies using KCR over architecting activities. KCR is widely employed in all architecting activities within the scope of this mapping study. We are not surprised by this result because the output of KCR provides the basis for using other knowledge-based approaches in architecting activities. For example, one of the prerequisites of using knowledge sharing, reasoning, and reuse is the existence of captured knowledge that is represented in certain forms. However, there are significant differences in the popularity of the KCR application in various architecting activities. Specifically, KCR is more popular in AE, AS, ADp, and AU, but is seldom employed in AIA and AA. This is partly because KCR is frequently used to capture and represent architectural knowledge for various purposes in AE, AS, ADp, and AU. Another reason is that ADp and AU require the architecture to be represented in a certain form for better description and further easier understanding. Finally, AIA and AA have received little attention in using knowledge-based approaches, thus KCR is seldom employed in these two architecting activities.

3.3. Architecting activities using knowledge-based approaches

This subsection elaborates on which architecting activities, knowledge-based approaches have been applied in.

Analyzing the map (Fig. 3) from the perspective of architecting activity dimension, we get the distribution of selected studies over architecting activities. As shown in Fig. 6, all the architecting activities (including general and specific architecting activities) employ knowledge-based approaches. However, significant differences in the study distribution exist over the ten architecting activities. Knowledge-based approaches are mostly used in AE with eighteen selected studies. In contrast, only two studies address the problems in AIA.

As revealed in Fig. 6, AIA is not a well-explored research area in architecting activities in terms of the application of knowledge-based approaches. *S4* proposes a meta-model of architectural design decisions to make the design decision knowledge explicit (a method of KCR), based on which an algorithm for AIA is designed to automatically identify all the architectural elements affected by a changed architectural design decision; *S17* employs KCR and KR to make explicit the architectural design decisions and their relationships, based on which the results of AIA are more reliable. These two studies both consider architectural design decisions as first-class architecture elements and perform AIA from the perspective of changing architectural design decisions. We argue that KS and KR should receive more attention in AIA, as the knowledge about AIA of existing architectures can facilitate the AIA of similar architectures (e.g., the architecture of software product lines). Consider a product line and concrete Products A and B. Performing AIA

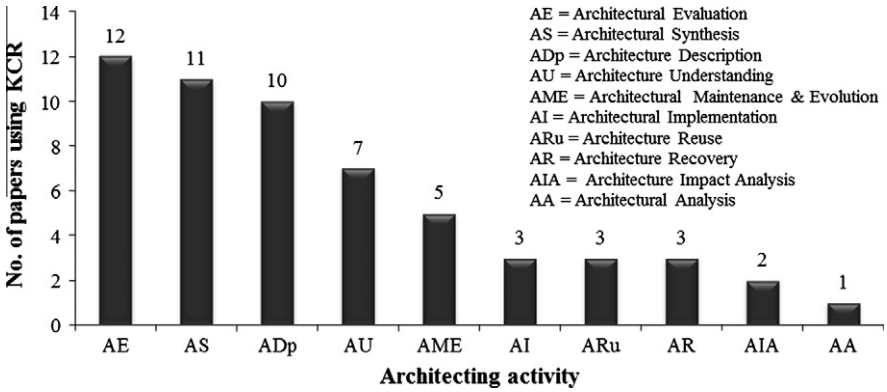

Fig. 5. Distribution of selected studies using KCR over architecting activities.

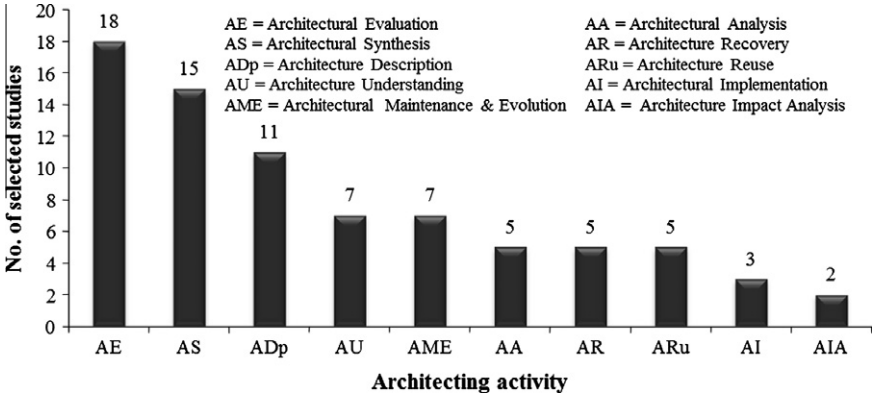

Fig. 6. Distribution of selected studies over architecting activities.

based on a requirement change in Product A can benefit from reusing AIA knowledge from Product B, and vice versa.

Architectural implementation (AI) also receives little attention on using knowledge-based approaches. There are only three studies in this area. Two (*S24* and *S41*) of them focus on object-oriented design and the third one (*S48*) concentrates on service-oriented architecture. To improve object-oriented design (i.e., architecture implementation in object-oriented design) using the accumulated knowledge systematically and effectively, *S24* proposed a method based on a defined rules catalog (that unifies various knowledge such as heuristics, principles, and bad smells) and the relationships between rules and patterns. *S41* represents the successful design experience in cases (a method of KCR) and employs Case-Based Reasoning approach to implement architectures in object-oriented design by reusing the cases. *S48* introduces an AK base on wireless services, as a means of sharing and reusing existing AK, to improve the quality of service-oriented architecture (SOA) implementation and speed up the SOA development with decreased cost. In the area of AI, the design and development communities have gained much experience on how to implement various architectures. Knowledge-based approaches can help designers and developers to capture, represent, share, and reuse their experiences on AI. For example, with knowledge-based approaches, designers are able to extract architecture patterns and pattern using context in the form of knowledge from past designs to facilitate the understanding and implementation of composing solutions (e.g., combination of patterns) that achieving specific quality attributes.

Researchers and practitioners have put most effort on AE and AS using knowledge-based approaches, with eighteen and fifteen selected studies, respectively. As mentioned in Section 2.1.1, AA, AS, and AE are the three main architecting activities in the model of architecture design proposed by Hofmeister et al. [15]. AS aims to propose architecture design alternatives for ASRs. AE makes sure that the selected architecture design is an appropriate one. These two architecting activities are essential to construct the final architecture. AA aims at defining the problems that an architecture should address with a special focus on identifying ASRs. However, AA received relatively less attention on using knowledge-based approaches with only five selected studies. This situation implies that knowledge-based approaches are considered important and useful to improve the practices in the solution space (i.e., AE and AS), but are not well-explored in terms of their application in the problem space (i.e., AA) of architecture design.

3.4. Study classifications by publication venue and year

This subsection gives details about which sources and in which years the selected studies of this mapping study have been published.

Table 9
Distribution of selected studies over publication types.

Publication Type	No.	%
Conference	25	45.45
Journal	17	30.91
Workshop	12	21.82
Book Chapter	1	1.82
Total	55	100.00

3.4.1. Classification by publication type and source

The selected studies were published in four publication types: Conference, Journal, Workshop, and Book Chapter. Table 9 shows the distribution of selected studies over publication types. Conference, Journal, and Workshop are three main publication types and they are with 45.45% (25 studies), 30.91% (17 studies), and 21.82% (12 studies) of the selected studies, respectively. Only one study was published as a Book Chapter. Table 10 presents the publication sources of all the selected studies, their types, number of studies, and their corresponding proportion against the total number of selected studies. Overall, 30 publication sources are identified which means this study topic has received wide attention in the SA research community. One observation is that there are one leading Workshop (SHARK), Conference (WICSA), and Journal (JSS) respectively as the publication venues for this study topic.

3.4.2. Classification by publication year

Fig. 7 presents the distribution of selected studies published from year 2000 to 2010. We did not plot the data of year 2011 in this figure because the data of 2011 is incomplete (covering only the studies of the first quarter). This distribution figure provides the SA community a clear message on the trend of the number of the published studies on knowledge-based approaches in SA. Overall, the number of selected studies in this topic has been increasing with little fluctuation from 2000 to 2010. This trend shows that the application of knowledge-based approaches is receiving increasing attention in SA community. Except year 2001, at least one study was published during the other years. Since 2004, there were at least four studies published each year. One reason for this could be that the software architecture community started a paradigm shift the year 2004 towards addressing architectural design decisions as first class entities in architecture design with their associated architectural knowledge (e.g., design rationale) [38,39].

3.5. Application domains

This subsection presents the application domains where knowledge-based approaches have been applied to, with a special focus

Table 10
Distribution of selected studies over publication sources.

Publication Source	Type	No.	%
SHAring and Reusing architectural Knowledge	Workshop	8	14.55
The Working IEEE/IFIP Conference on Software Architecture	Conference	7	12.73
Journal of Systems and Software	Journal	5	9.09
European Conference on Software Architecture	Conference	3	5.45
International Conference on the Quality of Software Architectures	Conference	3	5.45
European Conference on Software Maintenance and Reengineering	Conference	2	3.64
IEEE Transactions on Software Engineering	Journal	2	3.64
IEEE Software	Journal	2	3.64
International Journal of Software Engineering and Knowledge Engineering	Journal	2	3.64
International Conference on Software Reuse	Conference	1	1.82
IEEE International Conference on Engineering of Complex Computer Systems	Conference	1	1.82
International conference on Advanced Communication Technology	Conference	1	1.82
European Conference on Model-Driven Architecture - Foundations and Applications	Conference	1	1.82
International Conference and Workshops on the Engineering of Computer-Based Systems	Conference	1	1.82
International Conference on Automated Software Engineering	Conference	1	1.82
Asia Pacific Software Engineering Conference	Conference	1	1.82
EUROMICRO Conference on Software Engineering and Advanced Applications	Conference	1	1.82
IEEE International Symposium on Web Site Evolution	Conference	1	1.82
International Conference on Product Focused Software Process Improvement	Conference	1	1.82
IBM Systems Journal	Journal	1	1.82
ACM SIGSOFT Software Engineering Notes	Journal	1	1.82
IEE Proceedings – Software	Journal	1	1.82
Information and Software Technology	Journal	1	1.82
Information Sciences	Journal	1	1.82
Software Quality Journal	Journal	1	1.82
ER workshops AOIS, BP-UML, CoMoGIS, eCOMO, and QoIS	Workshop	1	1.82
Managing Requirements Knowledge	Workshop	1	1.82
International Workshop on Software and Performance	Workshop	1	1.82
IEEE International Workshop on Program Comprehension	Workshop	1	1.82
Collaborative Software Engineering	Book Chapter	1	1.82
Total		55	100.00

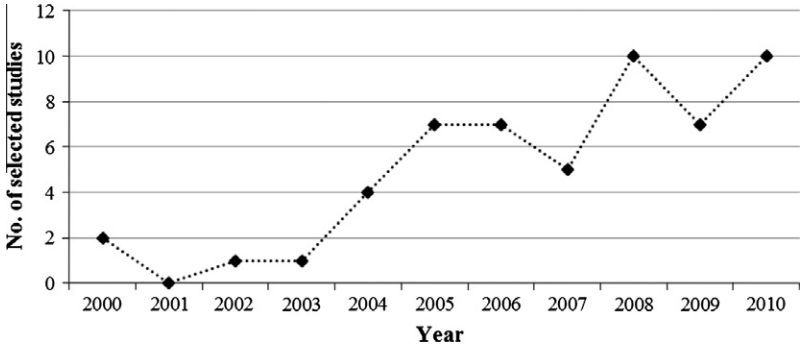

Fig. 7. Distribution of selected studies over time period.

on the studies with industrial evidence, i.e., the evidence obtained from industrial practice or industrial cases.

As shown in Table 11, 18 studies did not specify explicitly the application domains and the rest of the studies are classified into 13 application domains. Note that, if a study does not provide related information about its application domain, we classify this study into the domain "not specified". The application domain "Embedded software" has received the most attention in applying knowledge-based approaches in SA. For instance, *S27* proposed a knowledge-based quality-driven architecture design and evaluation approach, which was validated in the architecture design and evaluation of a secure middleware for embedded peer-to-peer systems.

To investigate industrial relevance, we further looked into which of the studies were industrial. We identified that 15 selected studies reach the evidence level 5 (i.e., evidence obtained from industrial studies) or above (i.e., the studies with score 0.8 and 1.0 in Q1 of Table 13). The application domain distribution of these studies is presented in Table 12. Studies with industrial evidence were conducted in eight application domains. The domain "Financial software" has received the most attention in applying knowledge-based approaches in industry. However, each domain has only few studies (i.e., no more than three).

4. Threats to validity

The results of this systematic mapping study may be influenced by the coverage of study search, bias on study selection, and inaccuracy in study data extraction. Therefore, four types of threats to the validity of the study results are discussed in the following subsections.

4.1. Conclusion validity

Conclusion validity (a.k.a. reliability) is concerned with whether the mapping study can be repeated by other researchers and get

Table 11
Distribution of application domains of the selected studies.

Application domain	No. of studies	Studies
Not specified	18	S5, S6, S19, S20, S21, S22, S24, S25, S28, S29, S32, S33, S34, S35, S38, S47, S53, S55
Embedded software	7	S14, S27, S30, S37, S41, S51, S54
E-commerce system	4	S2, S9, S18, S39
Information management system	3	S1, S26, S36
Financial software	3	S3, S44, S52
Aerospace software	3	S8, S11, S49
Astronomy software	3	S12, S17, S42
Automatic control software	3	S7, S13, S31
Web server software	2	S4, S23
Distributed system	2	S10, S50
Education software	2	S15, S43
Image processing software	2	S40, S46
Mobile software	2	S45, S48
Database management system	1	S16
Total	55	

Table 12
Distribution of application domains of the studies with industrial evidence.

Application domain	No. of studies	Studies
Financial software	3	S3, S44, S52
Embedded software	2	S27, S51
E-commerce system	2	S9, S39
Astronomy software	2	S17, S42
Automatic control software	2	S7, S13
Mobile software	2	S45, S48
Information management system	1	S26
Aerospace software	1	S11
Total	15	

the same results [40]. To make the results of this mapping study reproducible, the search terms used in automatic search and the search sources for both automatic and manual searches are presented. Moreover, inclusion and exclusion criteria for study selection are also defined. However, different researchers tend to have different understandings on these criteria, thus the results of study selection performed by various researchers are likely to be varied. To reduce the bias on study selection results, a pilot selection was performed to ensure that the reviewers reached a consensus on understanding the criteria. Also, the study protocol was discussed among researchers to ensure a common understanding on study selection. Furthermore, in the second and final round study selections, two reviewers conducted the selection process in parallel and independently, and then harmonized their selection results to mitigate the personal bias in study selection caused by individual reviewers.

The inaccuracy of the data extraction results may negatively affect the classification results of the selected studies. The quality

Table 13
Quality assessment results of selected studies.

Paper index	Q1	Q2	Q3	Q4	Q5	Total score	Paper index	Q1	Q2	Q3	Q4	Q5	Total score
S1	0.2	1.0	1.0	1.0	0.0	3.2	S29	0.2	1.0	0.0	1.0	0.0	2.2
S2	0.2	1.0	1.0	0.5	0.0	2.7	S30	0.6	1.0	1.0	1.0	0.5	4.1
S3	0.8	1.0	0.5	1.0	0.0	3.3	S31	0.2	1.0	0.5	0.5	0.0	2.2
S4	0.2	1.0	0.0	1.0	0.0	2.2	S32	0.2	1.0	1.0	1.0	0.0	3.2
S5	0.2	1.0	0.5	1.0	0.0	2.7	S33	0.2	1.0	1.0	1.0	0.0	3.2
S6	0.2	1.0	1.0	0.5	0.0	2.7	S34	0.0	1.0	0.5	1.0	0.0	2.5
S7	0.8	1.0	1.0	1.0	1.0	4.8	S35	0.0	1.0	1.0	1.0	1.0	4.0
S8	0.6	1.0	0.5	1.0	1.0	4.1	S36	0.6	1.0	1.0	1.0	0.0	3.6
S9	0.8	1.0	0.5	0.5	0.0	2.8	S37	0.2	1.0	1.0	1.0	0.0	3.2
S10	0.6	1.0	1.0	1.0	1.0	4.6	S38	0.6	1.0	1.0	1.0	1.0	4.6
S11	0.8	1.0	1.0	1.0	0.0	3.8	S39	0.8	1.0	1.0	1.0	0.5	4.3
S12	0.2	1.0	1.0	1.0	0.0	3.2	S40	0.2	1.0	1.0	1.0	1.0	4.2
S13	0.8	1.0	0.0	1.0	0.0	2.8	S41	0.6	1.0	1.0	1.0	0.5	4.1
S14	0.2	1.0	1.0	0.5	0.0	2.7	S42	0.8	1.0	1.0	1.0	0.0	3.8
S15	0.6	1.0	1.0	1.0	0.5	4.1	S43	0.2	1.0	1.0	1.0	0.0	3.2
S16	0.6	1.0	1.0	1.0	0.0	3.6	S44	0.8	1.0	1.0	1.0	0.5	4.3
S17	0.8	1.0	1.0	1.0	1.0	4.8	S45	0.8	1.0	1.0	1.0	0.0	3.8
S18	0.2	1.0	1.0	1.0	1.0	4.2	S46	0.4	1.0	1.0	1.0	0.0	3.4
S19	0.2	1.0	0.5	1.0	0.0	2.7	S47	0.2	1.0	1.0	1.0	0.0	3.2
S20	0.2	1.0	0.5	1.0	0.0	2.7	S48	0.8	1.0	1.0	0.5	0.5	3.8
S21	0.2	1.0	1.0	1.0	0.0	3.2	S49	0.6	1.0	1.0	1.0	0.5	4.1
S22	0.0	1.0	1.0	1.0	0.0	3.0	S50	0.6	1.0	1.0	1.0	0.0	3.6
S23	0.2	1.0	0.5	1.0	0.0	2.7	S51	0.8	1.0	1.0	1.0	0.0	3.8
S24	0.6	1.0	1.0	0.5	0.0	3.1	S52	1.0	1.0	1.0	1.0	0.0	4.0
S25	0.2	1.0	1.0	1.0	1.0	4.2	S53	0.0	1.0	0.5	1.0	0.0	2.5
S26	0.8	1.0	1.0	1.0	0.0	3.8	S54	0.6	1.0	1.0	1.0	0.5	4.1
S27	0.8	1.0	1.0	1.0	0.0	3.8	S55	0.2	1.0	1.0	1.0	0.0	3.2
S28	0.0	1.0	1.0	1.0	0.0	3.0							

	Q1	Q2	Q3	Q4	Q5	Total
Mean score	0.44	1.00	0.85	0.94	0.24	3.47

assessment on the selected studies helps to increase the accuracy of the data extraction results. The quality assessment results of the selected studies are shown in Table 13 according to the assessment questions listed in Table 7. The mean score of all the selected studies is 3.47. With that score, we can conclude that the overall quality of the selected studies is acceptable. The mean score of Q1 is 0.44, which results from 5 studies with score 0.0 (no evidence) and 22 studies with score 0.2 (evidence obtained from demonstration or working out toy examples). In other words, almost half of the selected studies have very low level of evidence. All the studies got full scores on question Q2. We paid special attention to questions Q3 and Q4 since their scores can reflect the accuracy of the data extraction results. The higher score of the study, the more accurate the data extraction results will be. Thus, the score provides a clue about whether the data extraction results of a study need to be checked more carefully. Each reviewer checked extracted data of all selected studies with special attention to the ones with low scores in Q3 and Q4, and reached a consensus when disagreement exists about the extracted data. Interestingly, the average score of Q5 is very low since 69.1% (i.e., 38 out of 55) of selected studies do not mention the limitations or drawbacks of their work.

4.2. Construct validity

Construct validity is concerned with whether the concepts being studied are interpreted correctly and whether the relevant studies can be collected completely [40]. In this mapping study, the architecting activities and knowledge-based approaches are the key concepts under consideration. To ensure the correct interpretation of these key concepts, we checked the definitions of the concepts with related literature and discussed these definitions among the authors to reach a consensus on the understanding of them. The correct interpretation of the concepts helps to extract the data precisely from the selected studies. In order to ensure the completeness of study search, we design a search strategy that employs a combination method of automatic search and manual search. Furthermore, to make sure the high coverage of potentially relevant studies in automatic search, we improved the search terms according to the results of the trial search before the formal search is performed. However, the refined list of search terms might not be comprehensive, some relevant papers, therefore, might be missing. To address this issue, manual search is used as complementary search to automatic search. We checked all the papers published in relevant journals, conferences, and workshops so that the potentially relevant studies published in these venues are included.

4.3. Internal validity

Internal validity focuses on the analysis of the extracted data [40]. Since the data analysis in this systematic mapping study only uses descriptive statistics, the threats to internal validity are minimal.

4.4. External validity

External validity is concerned with the representativeness of the selected studies regarding the overall goal of the mapping study [40]. The results of this mapping study were considered regarding the knowledge-based approaches in the SA domain. Therefore, the presented classification and systematic map of the selected studies and the conclusions drawn are only valid in the study topic. The predefined protocol is helpful to obtain a representative collection of studies in the given study topic. The representative venues for manual search also contribute to improving the representativeness of the selected studies. We did not include studies published before 2000, which does not affect the representativeness of the selected studies. This decision is supported by a systematic literature review on KM in SE showing that few related studies were published before 2000, and none of them focused on applying knowledge-based approaches in software architecting [5].

5. Discussion

This mapping study indicates that the application of knowledge-based approaches in SA is a quite immature area in both research and practice. First, more than two thirds of the selected studies (37 studies out of 55) are published in conferences and workshops, and only 30.9% (17 out of 55) of the selected studies have reached the maturity of a journal publication. Furthermore, only 27.3% (i.e., 15 out of 55) of the selected studies reach the evidence level 5 (i.e., evidence obtained from industrial studies) or above. In particular, only one study (S52) provides evidence obtained from industrial practice (i.e., the evidence level 6). Finally, almost half (49.1%, 27 out of 55) of the selected studies fall into the evidence level 2 (i.e., evidence obtained from demonstration or working out toy examples) or below. Especially, five selected studies do not provide any evidence.

The increasing number of selected studies over the last decade shows that the application of knowledge-based approaches is receiving increasing attention from the SA research community. The selected studies are published in 30 different venues, indicating that extensive attention on this study topic is being paid from researchers with a broad range of different research interests in SA. These two facts indicate that this study topic is likely to remain attractive. However we would urge the research community to strive for high-level evidence in future studies.

The results of this systematic mapping study also point out a number of implications for further research in SA:

(1) The results of this mapping study call for more investigation on improving AIA with knowledge-based approaches. AIA is an architecting activity that is frequently performed to determine which architecture elements are affected when a change scenario happens. The associated architectural knowledge on this change scenario, such as related design decisions and dependencies to these decisions, should be considered and employed to improve AIA.
(2) This mapping study also shows that the application of knowledge recovery approach in various forms needs to be explored seriously. In many architecting cases, architects need to recover the knowledge about architecture design, especially when maintaining or evolving the architecture that is not well described and documented. But little work has been done on the application of knowledge recovery in architecting activities.
(3) As discussed in Section 3.3, AI can benefit from knowledge-based approaches, e.g., knowledge sharing. However, little effort has been made in this area. AI is a knowledge-intensive activity, in which one architecture design can be implemented in a variety of detailed designs by different designers. Knowledge-based approaches can be used to capture the knowledge in various designs, share them with other designers, and reuse them according to different contexts. This research area has a clear need for more investigation.
(4) Little work has been done to provide automatic and semi-automatic knowledge reasoning to support architecting activities. Knowledge reasoning plays an important role in architecting activities, for example, it can be used to justify if a design decision is an appropriate one to address an

ASR [41]. However, knowledge reasoning usually requires considerable upfront effort to formally represent the knowledge, which is a prerequisite for automatic and semi-automatic knowledge reasoning. When knowledge reasoning can be (semi)-automated in architecting activities, the efficiency of architecture design, maintenance, and evolution will be significantly improved.

(5) More high level evidence-based research using knowledge-based approaches in architecting activities is needed. The overall evidence level of all the selected studies is relatively low, i.e., only 15 out of the total 55 selected studies reach the evidence level 5 (i.e., evidence obtained from industrial studies) or above. Although the research community is increasingly aware of the benefits of using knowledge-based approaches in software architecture, low-evidence-level studies cannot provide architects sufficient confidence to have a try. Therefore, there is an urgent need of high-evidence-level studies showing the real benefits that architect practitioners can get from knowledge-based approaches.

(6) Although knowledge-based approaches have been applied to the architectures of a wide range of application domains, there are still some application domains in which knowledge-based approaches are not or seldom employed so far (at least according to the results of this mapping study), e.g., telecommunications software, healthcare software, and cloud software. The software systems of such domains are often large-scale and complex. For instance, in the telecommunications domain, the software system of a typical wireless base station has millions of lines of source code and dozens of subsystems. Knowledge-based approaches, such as knowledge sharing and reuse, can facilitate conveying architectural knowledge to stakeholders and the reuse of the existing architectural knowledge of a specific domain.

6. Conclusions

We got 16,144 papers returned from the literature searches including the papers from both manual and automatic search, of which 55 were finally selected for data extraction. Based on the extracted data, we get an overview on what knowledge-based approaches are currently employed in which architecting activities and on the popularity of various knowledge-based approaches used in general and specific architecting activities. We draw conclusions around four main points:

(1) Knowledge Capture and Representation (KCR), Knowledge Reuse (KR), Knowledge Sharing (KS), Knowledge Reasoning (KRs), and Knowledge Recovery (KRv) are all used in the architecting activities, but there are significant differences on the popularity of these knowledge-based approaches. KCR is the most popular approach that is widely used in all the architecting activities. KRv is almost ignored as it is seldom employed in the architecting activities. More investigation needs to be performed to improve architecting activities using KRv.

(2) Knowledge-based approaches are used in all ten architecting activities. However, the study distribution is highly uneven over the ten activities. Comparing with other architecting activities, AIA receives the least attention on using knowledge-based approaches. This research area needs to be further explored. Also, AI using knowledge-based approaches is a promising research area and requires more research effort.

(3) 30 publication sources from journals, conferences, and workshops are identified, which means the study topic, application of knowledge-based approaches in SA, has received wide attention from the SA community. In addition, there are one leading Workshop (SHARK), Conference (WICSA), and Journal (JSS) respectively as the publication venues for this study topic. The trend on the number of published studies in each year indicates that researchers have made increasingly effort in this topic during the last decade.

(4) Knowledge-based approaches are applied to the software architecture of a wide range of application domains, and the domain "embedded software" has received the most attention. Industrial application and cases were conducted in eight application domains, but each domain has only few (no more than three) studies.

With the implications discussed in Section 5, we encourage the researchers and practitioners in SA community to carry out more empirical studies with high-level evidence on applying various knowledge-based approaches in the architecting process, in order to build a solid foundation for improving architecting activities with the support of knowledge-based approaches.

Acknowledgements

This work is partially supported by AFR-Luxembourg under the Contract No. 895528, and the NSFC under the Grant No. 60903034, QuASAK: Quality Assurance in Software architecting process using Architectural Knowledge. We would like to thank Matthias Galster for his valuable input and the anonymous reviewers for their constructive comments on the earlier version of this paper.

Appendix A. Selected studies

[S1] I. Ivkovic, K. Kontogiannis, A framework for software architecture refactoring using model transformations and semantic annotations, in: Proceedings of the 10th European Conference on Software Maintenance and Reengineering (CSMR'06), Bari, Italy, 2006, pp. 135–144.

[S2] M.C. Carignano, S. Gonnet, H. Leone, A model to represent architectural design rationale, in: Proceedings of the 8th Joint Working IEEE/IFIP Conference on Software Architecture and the 3rd European Conference on Software Architecture (WICSA/ECSA'09), Cambridge, UK, 2009, pp. 301–304.

[S3] A. Tang, Y. Jin, J. Han, A rationale-based architecture model for design traceability and reasoning, Journal of Systems and Software, 80 (6) (2007) 918–934.

[S4] Y. Choi, H. Choi, M. Oh, An architectural design decision-centric approach to architectural evolution, in: Proceedings of the 11th international conference on Advanced Communication Technology (ICACT'09), IEEE Press, Gangwon-Do, South Korea, 2009, pp. 417–422.

[S5] A. Erfanian, F.S. Aliee, An ontology-driven software architecture evaluation method, in: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (SHARK'08), ACM, Leipzig, Germany, 2008, pp. 79–86.

[S6] O. Zimmermann, Architectural Decisions as Reusable Design Assets, IEEE Software, 28 (1) (2011) 64–69.

[S7] D. Dhungana, R. Rabiser, P. Grunbaclier, H. Prahofer, C. Federspiel, K. Lehner, Architectural Knowledge in Product Line Engineering: An Industrial Case Study, in: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'06), Cavtat/Dubrovnik, Croatia, 2006, pp. 186–197.

[S8] C. Xiaofeng, S. Yanchun, X. Sai, M. Hong, Architecture Design for the Large-Scale Software-Intensive Systems: A Decision-Oriented Approach and the Experience, in: Proceedings of 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'09), Potsdam, Germany, 2009, pp. 30–39.

[S9] A. Tang, J. Han, Architecture Rationalization: A Methodology for Architecture Verifiability, Traceability and Completeness, in: Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS'05), IEEE Computer Society, Greenbelt, MD, USA, 2005, pp. 135–144.
[S10] A. Vasconcelos, C. Werner, Architecture Recovery and Evaluation Aiming at Program Understanding and Reuse, in: S. Overhage, C. Szyperski, R. Reussner, J. Stafford (Eds.) Proceedings of the 3rd international conference on Quality of software architectures (QoSA'07), Springer Berlin / Heidelberg, Medford, MA, USA, 2007, pp. 72–89.
[S11] M.A. Babar, R. Capilla, Capturing and Using Quality Attributes Knowledge in Software Architecture Evaluation Process, in: Proceedings of the 1st International Workshop on Managing Requirements Knowledge (MARK'08), IEEE Computer Society, Barcelona, Spain, 2008, pp. 53–62.
[S12] P. Liang, A. Jansen, P. Avgeriou, Collaborative Software Architecting Through Knowledge Sharing, in: I. Mistrík, A. van der Hoek, J. Grundy, J. Whitehead (Eds.) Collaborative Software Engineering, Springer Berlin Heidelberg, 2010, pp. 343–367.
[S13] D. Dhungana, R. Rabiser, P. Grunbacher, Decision-Oriented Modeling of Product Line Architectures, in: Proceedings of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA'07), Mumbai, Maharashtra, India, 2007, pp. 22–22.
[S14] F. Bachmann, L. Bass, M. Klein, C. Shelton, Designing software architectures to achieve quality attribute requirements, IEE Proceedings - Software, 152 (4) (2005) 153–165.
[S15] A. Jansen, J. Bosch, P. Avgeriou, Documenting after the fact: recovering architectural design decisions, Journal of Systems and Software, 81 (4) (2008) 536–557.
[S16] I. Ivkovic, M. Godfrey, Enhancing domain-specific software architecture recovery, in: Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03), IEEE Computer Society, Portland, Oregon, USA, 2003, pp. 266–273.
[S17] A. Jansen, P. Avgeriou, J.S.v.d. Ven, Enriching software architecture documentation, Journal of System and Software, 82 (8) (2009) 1232–1248.
[S18] J. Noppen, D. Tamzalit, ETAK: tailoring architectural evolution by (re-)using architectural knowledge, in: Proceedings of the 5th Workshop on Sharing and Reusing Architectural Knowledge (SHARK'10), ACM, Cape Town, South Africa, 2010, pp. 21–28.
[S19] A. Jansen, J. Bosch, Evaluation of tool support for architectural evolution, in: Proceedings of the 19th IEEE international conference on Automated Software Engineering (ASE'04), IEEE Computer Society, Linz, Austria, 2004, pp. 375–378.
[S20] O.L. Goaer, D. Tamzalit, M.C. Oussalah, A.-D. Seriai, Evolution styles to the rescue of architectural evolution knowledge, in: Proceedings of the 3rd international workshop on Sharing and reusing architectural Knowledge (SHARK'08), ACM, Leipzig, Germany, 2008, pp. 31–36.
[S21] C.A. Mattmann, D. Woollard, N. Medvidovic, Exploiting connector knowledge to efficiently disseminate highly voluminous data sets, in: Proceedings of the 3rd international workshop on Sharing and reusing architectural Knowledge (SHARK'08), ACM, Leipzig, Germany, 2008, pp. 37–40.
[S22] S. Trujillo, M. Azanza, O. Diaz, R. Capilla, Exploring extensibility of architectural design decisions, in: Proceedings of the 2nd Workshop on SHAring and Reusing architectural Knowledge (SHARK'07), IEEE Computer Society, Minneapolis, MN, USA, 2007, pp. 1–10.
[S23] C. Sylvain, Extraction of component-based architecture from object-oriented systems, in: S. Abdelhak, O. Mourad, T. Dalila (Eds.) Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA'08) Vancouver, BC, Canada, 2008, pp. 285–288.
[S24] J. Garzás, M. Piattini, Improving object-oriented micro architectural design through knowledge systematization, in: J. Akoka, S. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. van den Heuvel, M. Kolp, J. Trujillo, C. Kop, H. Mayr (Eds.) Perspectives in Conceptual Modeling, Springer Berlin/Heidelberg, 2005, pp. 444–453.
[S25] M. Shahin, P. Liang, M.R. Khayyambashi, Improving understandability of architecture design through visualization of architectural design decision, in: Proceedings of the 5th Workshop on Sharing and Reusing Architectural Knowledge (SHARK'10), ACM, Cape Town, South Africa, 2010, pp. 88–95.
[S26] R. Weinreich, G. Buchgeher, Integrating requirements and design decisions in architecture representation, in: Proceedings of the 4th European Conference on Software Architecture (ECSA'10), Springer-Verlag, Copenhagen, Denmark, 2010, pp. 86–101.
[S27] E. Ovaska, A. Evesti, K. Henttonen, M. Palviainen, P. Aho, Knowledge based quality-driven architecture design and evaluation, Information and Software Technology, 52 (6) (2010) 577–601.
[S28] M. Ramachandran, D. Mangano, Knowledge based reasoning for software architectural design strategies, SIGSOFT Software Engineering Notes, 29 (3) (2004) 1–4.
[S29] S. Khajenoori, L. Prem, K. Stevens, B.S. Keng, N. Kameli, Knowledge centered assessment pattern: an effective tool for assessing safety concerns in software architecture, Journal of System and Software, 73 (2) (2004) 313–322.
[S30] M. Kevin L., G. Hassan, Knowledge-based automation of a design method for concurrent systems, IEEE Transactions on Software Engineering, 28 (3) (2002) 228–255.
[S31] W. Wu, T. Kelly, Managing architectural design decisions for safety–critical software systems, in: C. Hofmeister, I. Crnkovic, R. Reussner (Eds.) Quality of Software Architectures, Springer Berlin/Heidelberg, 2006, pp. 59–77.
[S32] A. Tang, H. van Vliet, Modeling constraints improves software architecture design reasoning, in: Proceedings of the 8th Joint Working IEEE/IFIP Conference on Software Architecture and the 3rd European Conference on Software Architecture (WICSA/ECSA'09), IEEE, Cambridge, UK, 2009, pp. 253–256.
[S33] A. Smeda, M. Oussalah, T. Khammaci, My architecture: a knowledge representation meta-model for software architecture, International Journal of Software Engineering and Knowledge Engineering, 18 (7) (2008) 877–894.
[S34] A.W. Kiwelekar, R.K. Joshi, Ontological analysis for generating baseline architectural descriptions, in: Proceedings of the 4th European conference on Software architecture (ECSA'10), Springer-Verlag, Copenhagen, Denmark, 2010, pp. 417–424.
[S35] C. Pahl, Ontology-based composition and transformation for model-driven service architecture, in: A. Rensink, J. Warmer (Eds.) Model Driven Architecture – Foundations and Applications, Springer Berlin/Heidelberg, 2006, pp. 198–212.
[S36] Z. Yonggang, R. Witte, J. Rilling, V. Haarslev, Ontology-based program comprehension tool supporting website architectural evolution, in: Proceedings of the 8th IEEE International Symposium on Web Site Evolution (WSE'06), Philadelphia, Pennsylvania, USA, 2006, pp. 41–49.
[S37] I. Lera, C. Juiz, R. Puigjaner, C. Kurz, G. Haring, J. Zottl, Performance assessment on ambient intelligent applications through ontologies, in: Proceedings of the 5th international workshop on Software and performance (WOSP'05), ACM, Palma, Illes Balears, Spain, 2005, pp. 205–216.
[S38] J.W. Keung, T. Nguyen, Quantitative analysis for non-linear system performance data using case-based reasoning, in: Proceedings of the 17th Asia Pacific Software Engineering

Conference (APSEC'10), IEEE Computer Society, Sydney, Australia, 2010, pp. 346–355.
[S39] R. Roeller, P. Lago, H.v. Vliet, Recovering architectural assumptions, Journal of System and Software 79 (4) (2006) 552–573.
[S40] P. Rossel, D. Perovich, M. Bastarrica, Reuse of architectural knowledge in SPL development, in: S. Edwards, G. Kulczycki (Eds.) Formal Foundations of Reuse and Domain Engineering, Springer Berlin/Heidelberg, 2009, pp. 191–200.
[S41] G. Vazquez, J. Andres Diaz Pace, M. Campo, Reusing design experiences to materialize software architectures into object-oriented designs, Information Sciences, (0) (2010).
[S42] A. Jansen, T. Vries, P. Avgeriou, M. Veelen, Sharing the architectural knowledge of quantitative analysis, in: Proceedings of the 4th International Conference on Quality of Software Architectures (QoSA'08), Springer-Verlag, Karlsruhe, Germany, 2008, pp. 220–234.
[S43] A. Jansen, J. Bosch, Software Architecture as a set of architectural design decisions, in: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE Computer Society, Pittsburgh, Pennsylvania, USA, 2005, pp. 109–120.
[S44] A. Tang, J. Han, R. Vasa, Software architecture design reasoning: a case for improved methodology support, IEEE Software, 26 (2) (2009) 43–49.
[S45] C.D. Rosso, A. Maccari, Assessing the architectonics of large, software-intensive systems using a knowledge-based approach, in: Proceedings of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA'07), IEEE Computer Society, Mumbai, Maharashtra, India, 2007, pp. 2.
[S46] I. Pashov, M. Riebisch, I. Philippow, Supporting architectural restructuring by analyzing feature models, in: Proceedings of the 18th Euromicro Working Conference on Software Maintenance and Reengineering (CSMR'04), IEEE Computer Society, Tampere, Finland, 2004, pp. 25–36.
[S47] M. Ali-Babar, The application of knowledge-sharing workspace paradigm for software architecture processes, in: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (SHARK'08), ACM, Leipzig, Germany, 2008, pp. 45–48.
[S48] E. Niemela, J. Kalaoja, P. Lago, Toward an architectural knowledge base for wireless service engineering, IEEE Transactions on Software Engineering, 31 (5) (2005) 361–379.
[S49] C. Xiaofeng, S. Yanchun, M. Hong, Towards automated solution synthesis and rationale capture in decision-centric architecture design, in: Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA'08) Vancouver, BC, Canada, 2008, pp. 221–230.
[S50] L. Zhu, A. Aurum, I. Gorton, R. Jeffery, Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process, Software Quality Control, 13 (4) (2005) 357–375.
[S51] V.-P. Eloranta, K. Koskimies, Using domain knowledge to boost software architecture evaluation, in: Proceedings of the 4th European conference on Software architecture (ECSA'10), Springer-Verlag, Copenhagen, Denmark, 2010, pp. 319–326.
[S52] A. Akerman, J. Tyree, Using ontology to support development of software architectures, IBM Systems Journal, 45 (4) (2006) 813–825.
[S53] W. Wang, J.E. Burge, Using rationale to support pattern-based architectural design, in: Proceedings of the 5th Workshop on Sharing and Reusing Architectural Knowledge (SHARK'10), ACM, Cape Town, South Africa, 2010, pp. 1–8.
[S54] L. Bratthall, E. Johansson, B. Regnell, Is a design rationale vital when predicting change impact? A controlled experiment on software architecture evolution, in: Proceedings of the 2nd International Conference on Product Focused Software Process Improvement (PROFES'00), Springer-Verlag, Oulu, Finland, 2000, pp. 126–139.
[S55] M.J. Gerken, Specification of software architecture, International Journal of Software Engineering and Knowledge Engineering, 10 (1) (2000) 69–95.

References

[1] P. Kruchten, H. Obbink, J. Stafford, The Past, Present, and Future for Software Architecture, IEEE Software 23 (2006) 22–30.
[2] M. Shaw, P. Clements, The golden age of software architecture, IEEE Software 23 (2006) 31–39.
[3] ISO, Systems and Software Engineering – Architecture Description, ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011, pp. 1–46.
[4] I. Rus, M. Lindvall, Knowledge management in software engineering, IEEE Software 19 (2002) 26–38.
[5] F.O. Bjørnson, T. Dingsøyr, Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used, Information and Software Technology 50 (2008) 1055–1068.
[6] E. Hudlicka, Requirements elicitation with indirect knowledge elicitation techniques: comparison of three methods, in: Proceedings of the 2nd International Conference on Requirements Engineering (ICRE'96), Colorado Springs, Colorado, USA, 1996, pp. 4–11.
[7] E. Ovaska, A. Evesti, K. Henttonen, M. Palviainen, P. Aho, Knowledge based quality-driven architecture design and evaluation, Information and Software Technology 52 (2010) 577–601.
[8] K.W. Ong, M.Y. Tang, Knowledge management approach in mobile software system testing, in: IEEE International Conference on Industrial Engineering and Engineering Management (IEEM'07), Singapore, 2007, pp. 2120–2123.
[9] A.W. Kiwelekar, R.K. Joshi, Ontological analysis for generating baseline architectural descriptions, in: Proceedings of the 4th European conference on Software architecture (ECSA'10), Springer-Verlag, Copenhagen, Denmark, 2010, pp. 417–424.
[10] P. Lago, P. Avgeriou, First workshop on sharing and reusing architectural knowledge, SIGSOFT Software Engineering Notes 31 (2006) 32–36.
[11] A. Jansen, J. Bosch, Software Architecture as a Set of Architectural Design Decisions, in: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE Computer Society, Pittsburgh, Pennsylvania, USA, 2005, pp. 109–120.
[12] P. Kruchten, P. Lago, H. van Vliet, Building up and reasoning about architectural knowledge, in: C. Hofmeister, I. Crnkovic, R. Reussner (Eds.), Quality of Software Architectures, Springer, Berlin/Heidelberg, 2006, pp. 43–58.
[13] P. Liang, P. Avgeriou, Tools and technologies for architecture knowledge management, in: M. Ali Babar, T. Dingsøyr, P. Lago, H. Vliet (Eds.), Software Architecture Knowledge Management, Springer, Berlin Heidelberg, 2009, pp. 91–111.
[14] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M. Ali Babar, A comparative study of architecture knowledge management tools, Journal of Systems and Software 83 (2010) 352–370.
[15] C. Hofmeister, P. Kruchten, R.L. Nord, H. Obbink, A. Ran, P. America, A general model of software architecture design derived from five industrial approaches, Journal of Systems and Software 80 (2007) 106–126.
[16] B.A. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3. EBSE Technical Report EBSE-2007-01, in, Keele University and Durham University, 2007.
[17] E. Engström, P. Runeson, Software product line testing – a systematic mapping study, Information and Software Technology 53 (2011) 2–13.
[18] ISO, International Standard – ISO/IEC 14764 IEEE Std 14764-2006 Software Engineering – Software Life Cycle Processes – Maintenance, ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998), 2006, 0_1-46.
[19] IEEE, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Std 1471-2000, 2000, i-23.
[20] M. Aoyama, Metrics and analysis of software architecture evolution with discontinuity, in: Proceedings of the 5th International Workshop on Principles of Software Evolution (IWPSE'02), ACM, Orlando, Florida, 2002, pp. 103–107.
[21] C. Riva, Reverse architecting: an industrial experience report, in: Proceedings of the 7th Working Conference on Reverse Engineering (WCRE'00), IEEE Computer Society, Brisbane, Queensland, Australia, 2000, pp. 42–50.
[22] P. Bengtsson, N. Lassing, J. Bosch, H. van Vliet, Architecture-level modifiability analysis (ALMA), Journal of Systems and Software 69 (2004) 129–147.
[23] IEEE, IEEE Standard for Information Technology – System and Software Life Cycle Processes – Reuse Processes, IEEE Std 1517-2010 (Revision of IEEE Std 1517-1999), 2010, pp. 1–51.
[24] Y. Merali, J. Davies, Knowledge capture and utilization in virtual communities, in: Proceedings of the 1st International Conference on Knowledge capture (K-CAP'01), ACM, Victoria, British Columbia, Canada, 2001, pp. 92–99.
[25] Ikujiro Nonaka, H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, USA, 1995.

[26] M. Alavi, D.E. Leidner, Review: knowledge management and knowledge management systems: conceptual foundations and research issues, MIS Quarterly 25 (2001) 107–136.
[27] L. Chen, M. Ali Babar, H. Zhang, Towards an evidence-based understanding of electronic data sources, in: Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering (EASE'10), Keele University, UK, 2010.
[28] B.A. Kitchenham, E. Mendes, G.H. Travassos, Cross versus within-company cost estimation studies: a systematic review, IEEE Transactions on Software Engineering 33 (2007) 316–329.
[29] V. Alves, N. Niu, C. Alves, G. Valença, Requirements engineering for software product lines: a systematic literature review, Information and Software Technology 52 (2010) 806–820.
[30] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Information and Software Technology 50 (2008) 833–859.
[31] M.S. Ali, M. Ali Babar, L. Chen, K.-J. Stol, A systematic review of comparative evidence of aspect-oriented programming, Information and Software Technology 52 (2010) 871–887.
[32] R. Farenhorst, R. Izaks, P. Lago, H.v. Vliet, A just-in-time architectural knowledge sharing portal, in: Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA'08), IEEE Computer Society, Vancouver, BC, Canada, 2008, pp. 125–134.
[33] K. Henttonen, M. Matinlassi, Open source based tools for sharing and reuse of software architectural knowledge, in: Proceedings of the 8th Joint Working IEEE/IFIP Conference on Software Architecture and the 3rd European Conference on Software Architecture (WICSA/ECSA'09), Cambridge, UK, 2009, pp. 41–50.
[34] R. Farenhorst, P. Lago, H. van Vliet, Effective tool support for architectural knowledge sharing, in: F. Oquendo (Ed.), Software Architecture, Springer, Berlin/Heidelberg, 2007, pp. 123–138.
[35] R. Farenhorst, H.v. Vliet, Understanding how to support architects in sharing knowledge, in: Proceedings of the 4th Workshop on Sharing and Reusing Architectural Knowledge (SHARK'09), IEEE Computer Society, Vancouver, Canada, 2009, pp. 17–24.
[36] P. Liang, A. Jansen, P. Avgeriou, Selecting a high-quality central model for sharing architectural knowledge, in: Proceedings of the 8th International Conference on Quality Software (QSIC '08), Oxford, UK, 2008, pp. 357–365.
[37] P. Liang, A. Jansen, P. Avgeriou, A. Tang, L. Xu, Advanced quality prediction model for software architectural knowledge sharing, Journal of System and Software 84 (2011) 786–802.
[38] J. Bosch, Software architecture: the next step, in: Proceedings of the 1st European Workshop on Software Architecture (EWSA'04), St. Andrews, UK, 2004, pp. 194–199.
[39] P. Kruchten, An ontology of architectural design decisions in software intensive systems, in: 2nd Groningen Workshop Software Variability, 2004, pp. 54–61.
[40] S. Easterbrook, J. Singer, M.-A. Storey, D. Damian, Selecting empirical methods for software engineering research, in: F. Shull, J. Singer, D.I.K. Sjøberg (Eds.), Guide to Advanced Empirical Software Engineering, Springer, London, 2008, pp. 285–311.
[41] A. Tang, J. Han, R. Vasa, Software architecture design reasoning: a case for improved methodology support, IEEE Software 26 (2009) 43–49.